REMARKS

In the final Office Action, the Examiner rejected claim 16 under 35 U.S.C. § 103(a) as unpatentable over Adya et al. (U.S. Patent Application Publication No. 2005/0102268) in view of Nitta et al. (U.S. Patent No. 5,287,521); and rejected claims 1-15 and 17 under 35 U.S.C. § 103(a) as unpatentable over Adya et al. in view of McClaughry et al. (U.S. Patent No. 5,933,825) and Nitta et al.

By this Amendment, Applicants propose amending claims 10 and 11 to change their dependency from claim 9 to claim 1. Applicants respectfully traverse the Examiner's rejections under 35 U.S.C. § 103. Claims 1-17 remain pending.

REJECTION UNDER 35 U.S.C. § 103 BASED ON ADYA ET AL. and NITTA ET AL.

In paragraph 3 of the final Office Action, the Examiner rejected claim 16 under 35 U.S.C. § 103(a) as allegedly unpatentable over Adya et al. and Nitta et al. Applicants respectfully traverse the rejection.

Independent claim 16 is directed to a method for performing first and second operations within a same directory. The method comprises obtaining a first lock on a sub-directory or file name within the directory by the first operation; obtaining a second lock on a sub-directory or file name within the directory by the second operation; determining whether the first and second locks conflict; concurrently performing the first and second operations when the first and second locks do not conflict, the first and second locks being read-write locks; and serializing performance of the first and second operations when the first and second locks conflict.

Neither Adya et al. nor Nitta et al., whether taken alone or in any reasonable combination, discloses or suggests the combination of features recited in claim 16. For example,

Adya et al. and Nitta et al. do not disclose or suggest determining whether a first lock obtained by a first operation conflicts with a second lock obtained by a second operation.

The Examiner alleged that Adya et al. discloses determining whether first and second locks conflict and cited paragraphs 0121-0123 of Adya et al. for support (final Office Action, page 2). Applicants respectfully disagree.

At paragraph 0121, Adya et al. discloses:

> Read Lock. The Read lock is requested by an application so that the application can read the associated file. The Read lock, in conjunction with the Write lock, allows the directory group to keep data in the file consistent.

In this paragraph, Adya et al. discloses a read lock that an application can request to read an associated file. Nowhere in this section, or elsewhere, does Adya et al. disclose or suggest determining whether a first lock obtained by a first operation conflicts with a second lock obtained by a second operation, as required by claim 16.

At paragraph 0122, Adya et al. discloses:

> Write Lock. The Write lock is requested by an application so that the application can write to (also referred to as update) the associated file. The Write lock, in conjunction with the Read lock, allows the directory group to keep data in the file consistent.

In this paragraph, Adya et al. discloses a write lock that an application can request to write to an associated file. Nowhere in this section, or elsewhere, does Adya et al. disclose or suggest determining whether a first lock obtained by a first operation conflicts with a second lock obtained by a second operation, as required by claim 16.

At paragraph 0123, Adya et al. discloses:

> When an application desires to open an object, the directory group performs two checks: (1) are the modes the application is asking for going to conflict with another application that has already opened the object; and (2) are the operations that the application is willing to share the object for going to conflict with what another application has already opened the object for and indicated it is willing to share the object for. Six of the ten

> locks are directed to supporting this checking: Open Read, Open Write, Open Delete, Open Not Shared Read, Open Not Shared Write, and Open Not Shared Delete. These locks are used to grant an application the ability to open an object, but do not necessarily guarantee that the data for the object can be obtained (the Read lock or Write lock (depending on the type of operation the application desires to perform) is obtained to access the data). Open Read Lock. The Open Read lock is requested by an application to allow the application to open the associated object for reading.

In this paragraph, Adya et al. discloses that an application desiring to open an object performs two checks: (1) whether the modes the application is asking for are going to conflict with another application that has already opened the object; and (2) whether the operations that the application is willing to share the object for are going to conflict with that which another application has indicated it is willing to share the object. Nowhere in this section, or elsewhere, does Adya et al. disclose or suggest determining whether a first lock obtained by a first operation conflicts with a second lock obtained by a second operation, as required by claim 16.

In other words, claim 16 recites obtaining the first and second locks, and then determining whether the locks conflict. Adya et al. specifically discloses that a request for a lock is denied when the locks needed by the request conflict with locks that have been granted to a different client (paras. 0133 and 0135). Therefore, Adya et al. cannot disclose or suggest determining whether a first lock obtained by a first operation conflicts with a second lock obtained by a second operation, as required by claim 16. Nitta et al. also does not disclose or suggest this feature.

Because Adya et al. and Nitta et al. do not disclose or suggest determining whether a first lock obtained by a first operation conflicts with a second lock obtained by a second operation, Adya et al. and Nitta et al. cannot disclose or suggest concurrently performing the first and second operations when the first and second locks are determined to not conflict, or serializing

performance of the first and second operations when the first and second locks are determined to conflict, as further recited in claim 16.

For at least these reasons, Applicants submit that claim 16 is patentable over Adya et al. and Nitta et al., whether taken alone or in any reasonable combination.

Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejection of claim 16 under 35 U.S.C. § 103 based on Adya et al. and Nitta et al.

REJECTION UNDER 35 U.S.C. § 103 BASED ON ADYA ET AL., MCCLAUGHRY ET AL., AND NITTA ET AL.

In paragraph 4 of the final Office Action, the Examiner rejected claims 1-15 and 17 under 35 U.S.C. § 103(a) as allegedly unpatentable over Adya et al. in view of McClaughry et al. and Nitta et al. Applicants respectfully traverse the rejection.

Independent claim 1, for example, is directed to a method for performing operations within a file system in which directories and files are organized as nodes in a namespace tree. The method comprises associating a read-write lock with each of the nodes in the namespace tree; acquiring a first lock on a name of one or more directories involved in a first operation; acquiring a second lock on an entire pathname involved in the first operation; determining whether the first lock or the second lock conflicts with third locks acquired by a second operation; performing the first operation when the first lock or the second lock does not conflict with the third locks, where the first, second, and third locks are read-write locks; and serializing performance of the first and second operations when the first lock or the second lock conflicts with the third locks.

Neither Adya et al., McClaughry et al., nor Nitta et al., whether taken alone or in any

reasonable combination, discloses or suggests the combination of features recited in claim 1. For example, Adya et al., McClaughry et al., and Nitta et al. do not disclose or suggest determining whether a first lock or a second lock acquired by a first operation conflicts with third locks acquired by a second operation.

For at least reasons similar to reasons given above with regard to claim 16, Adya et al. and Nitta et al. do not disclose or suggest this feature. McClaughry et al. also does not disclose or suggest this feature. For example, McClaughry et al. discloses denying a request for a lock when a conflict exists and notifying the client (col. 3, lines 14-17; col. 5, lines 53-58; and col. 6, lines 42-46). Therefore, Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination, do not disclose or suggest determining whether a first lock or a second lock acquired by a first operation conflicts with third locks acquired by a second operation, as required by claim 1.

Because Adya et al., McClaughry et al., and Nitta et al. do not disclose or suggest determining whether a first lock or a second lock acquired by a first operation conflicts with third locks acquired by a second operation, Adya et al., McClaughry et al., and Nitta et al. cannot disclose or suggest performing the first operation when the first lock or the second lock is determined to not conflict with the third locks, or serializing performance of the first and second operations when the first lock or the second lock is determined to conflict with the third locks, as further recited in claim 1.

For at least these reasons, Applicants submit that claim 1 is patentable over Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination. Claims 2-12 depend from claim 1 and are, therefore, patentable over Adya et al., McClaughry et

al., and Nitta et al. for at least the reasons given with regard to claim 1. Claims 2-12 are also patentable over Adya et al., McClaughry et al., and Nitta et al. for reasons of their own.

For example, claim 8 recites using a lazily allocated data structure that maps pathnames to locks to determine whether the first lock or the second lock conflicts with the third locks. Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination, do not disclose or suggest the feature recited in claim 8.

The Examiner alleged that Adya et al. discloses the feature of claim 8 and cited paragraphs 0046, 0102-0106, 0115, 0123, and 0131-0133 of Adya et al. for support (final Office Action, page 6). Applicants respectfully disagree.

At paragraph 0046, Adya et al. discloses:

> Generally, according to exclusive encryption, a plaintext name (the file or directory name within the directory entry) is mapped to a new name. The mapped name is optionally decasified into a decasified (case-insensitive) name and corresponding case information, allowing duplicate name detection to be case-insensitive. The mapped (and optionally decasified) name is then encoded and encrypted. This encrypted name (and optionally accompanying case information) are forwarded to the directory group that is responsible for managing the directory entry (e.g., based on pathname, as discussed in more detail below).

In this section, Adya et al. discloses that a file or directory name is mapped to a new name that is optionally made case-insensitive. Nowhere in this section, or elsewhere, does Adya et al. disclose or remotely suggest using a lazily allocated data structure that maps pathnames to locks to determine whether the first lock or the second lock conflicts with the third locks, as required by claim 8.

At paragraphs 0102-0106, Adya et al. discloses that each computer in the file system can maintain a local cache that maps a subset of the pathnames in the name space to the directory group that manages that pathname. Nowhere in this section, or elsewhere, does Adya et al.

disclose or remotely suggest using a lazily allocated data structure that maps pathnames to locks to determine whether the first lock or the second lock conflicts with the third locks, as required by claim 8.

At paragraph 0115, Adya et al. discloses:

> FIG. 8 is a flowchart illustrating an exemplary process for storing a file in a serverless distributed file system. Initially, a new file storage request is received at a client computing device (act 802). The client encrypts the file and the file name and generates the file contents hash (act 804). The client sends the encrypted file name and file contents hash to the appropriate Byzantine.-fault-tolerant directory group along with a request to create a directory entry (act 806). The directory group validates the request (act 808), such as by verifying that the file name does not conflict with an existing name and that the client has permission to do what it is requesting to do. If the request is not validated then the request fails (act 810). However, if the request is validated, then the directory group generates a directory entry for the new file (act 812). The directory group also determines the replica set for the new file and adds the replica set to the newly generated directory entry (act 814). Replicas of the file are also generated (act 816), and saved to multiple computers in the file system (act 818).

In this section, Adya et al. discloses that a client encrypts a file and a file name and generates a file contents hash and sends this information to a directory group that validates the request. Nowhere in this section, or elsewhere, does Adya et al. disclose or remotely suggest using a lazily allocated data structure that maps pathnames to locks to determine whether the first lock or the second lock conflicts with the third locks, as required by claim 8.

Paragraph 0123 of Adya et al. is reproduced above. In paragraph 0123, Adya et al. discloses that when an application requests to open an object, it is determined whether the modes that the application is requesting conflict with another application that has already opened the object and whether the operations for which the application is willing to share conflict with another application that has already opened the object. Nowhere in this section, or elsewhere, does Adya et al. disclose or remotely suggest using a lazily allocated data structure that maps pathnames to locks to determine whether the first lock or the second lock conflicts with the third

locks, as required by claim 8.

At paragraphs 0131-0133, Adya et al. discloses:

> Exclusive Lock. The Exclusive lock is requested by an application to obtain all of the previously discussed nine locks, including an Insert lock on each possible name that could exist (but does not already exist) in the directory. An Exclusive lock on a directory does not imply Exclusive locks on the files or subdirectories in the directory, but rather only on the directory's namespace. The Exclusive lock conflicts with each of the previously discussed nine locks.
>
> Various conflicts exist between the various different locks. Table I is a conflict matrix illustrating the conflicts between locks in one exemplary implementation. The following abbreviations are used in Table I: Ins (Insert), Excl (Exclusive), O-R (Open Read), O-W (Open Write), O-D (Open Delete), O-!R (Open Not Shared Read), O-!W (Open Not Shared Write), and O-!D (Open Not Shared Delete). An "X" in a cell of Table I indicates a conflict between the corresponding two locks-for example, Open Read conflicts with Open Not Shared Read but does not conflict with Open Not Shared Write.
>
> FIG. 9 is a flowchart illustrating an exemplary process for determining whether to allow a particular object to be opened. The process of FIG. 9 is implemented by the directory group responsible for managing the particular object. In the process of FIG. 9, it is assumed that the client requesting to open the particular object does not already have the necessary lock(s) to open the object as desired. Initially, a request to access an object with particular locks identified is received (act 902). A check is made by the directory group as to whether the modes implied by the selected locks conflict with locks that have been granted to a different client (act 904). For example, if the request is a request to open an object for reading, but another application has already opened the object with the Not Shared Read lock, then the mode (open read) implied by the selected lock conflicts with another application that has already opened the object. Because the directory group knows only if it has issued a conflicting lock to a client, but not whether the client is currently using the lock to allow an application access to an object, in some cases making the check in act 904 requires asking a client that currently holds a lock is willing to give it up.

In this section, Adya et al. discloses information regarding an exclusive lock, information regarding the various conflicts that exist between the various different locks, and that a check is made whether the modes implied by locks requested by a client conflict with locks that have been granted to a different client. Nowhere in this section, or elsewhere, does Adya et al. disclose or remotely suggest using a lazily allocated data structure that maps pathnames to locks to determine whether the first lock or the second lock conflicts with the third locks, as required

by claim 8.

McClaughry et al. and Nitta et al. also do not disclose or suggest this feature of claim 8.

For at least these additional reasons, Applicants submit that claim 8 is patentable over Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination.

Claim 10 recites determining an order for the first, second, and third locks based on levels of the namespace tree involved in the first, second, and third locks and within one of the levels of the namespace tree involved in the first, second, and third locks. Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination, do not disclose or suggest the features of claim 10.

The Examiner alleged that Adya et al. discloses the features of claim 10 and cited paragraphs 0115-0118 of Adya et al. for support (final Office Action, page 6). Applicants respectfully disagree.

At paragraph 0115, Adya et al. discloses:

> FIG. 8 is a flowchart illustrating an exemplary process for storing a file in a serverless distributed file system. Initially, a new file storage request is received at a client computing device (act 802). The client encrypts the file and the file name and generates the file contents hash (act 804). The client sends the encrypted file name and file contents hash to the appropriate Byzantine.-fault-tolerant directory group along with a request to create a directory entry (act 806). The directory group validates the request (act 808), such as by verifying that the file name does not conflict with an existing name and that the client has permission to do what it is requesting to do. If the request is not validated then the request fails (act 810). However, if the request is validated, then the directory group generates a directory entry for the new file (act 812). The directory group also determines the replica set for the new file and adds the replica set to the newly generated directory entry (act 814). Replicas of the file are also generated (act 816), and saved to multiple computers in the file system (act 818).

In this paragraph, Adya et al. discloses steps involved in storing a file in a distributed file system. Nowhere in this section, or elsewhere, does Adya et al. disclose or suggest determining

an order for first, second, and third locks, let alone determining an order based on levels of the namespace tree involved in the first, second, and third locks and within one of the levels of the namespace tree involved in the first, second, and third locks, as required by claim 10.

At paragraph 0116, Adya et al. discloses:

> By storing the directory entries in a Byzantine group, and including file verification data in the entries, fault tolerance is maintained (up to .function. failures). However, storage space requirements and Byzantine operations are reduced by storing files separately from directories and not using Byzantine operations to access them. For example, directory entries may be on the order of one hundred bytes, whereas the file itself may be on the order of thousands or even billions of bytes.

In this paragraph, Adya et al. discloses that directory entries are stored in a Byzantine group, and files are stored separately from directories. Nowhere in this section, or elsewhere, does Adya et al. disclose or suggest determining an order for first, second, and third locks, let alone determining an order based on levels of the namespace tree involved in the first, second, and third locks and within one of the levels of the namespace tree involved in the first, second, and third locks, as required by claim 10.

At paragraphs 0117 and 0118, Adya et al. discloses:

> Directory and File Lock Mechanism
>
> Each object (e.g., directory and file) in distributed file system 150 of FIG. 1 has associated with it a set of leased locks. These locks are used to determine, based on the type of operation an application desires to perform, whether the application can open a directory or file to perform that operation. A lock can be viewed as a lease with a particular time span that depends on the type of lock and the level of contention. For example, the time span on a write lock may be a few minutes, while the time span on a read lock may be as long as a few days. When an application desires to perform an operation(s) on an object, the client computer on which the application is executing looks to see if it already has the necessary locks to perform the operation(s). If not, it requests the appropriate lock(s) from the directory group responsible for managing that object. Once the application has finished performing the desired operation, it can optionally release the lock(s) it acquired or keep it until it automatically expires or is recalled by the managing directory group.

In these paragraphs, Adya et al. discloses leased locks that are used to determine whether an

application can open a directory or file to perform an operation. Nowhere in this section, or elsewhere, does Adya et al. disclose or suggest determining an order for first, second, and third locks, let alone determining an order based on levels of the namespace tree involved in the first, second, and third locks and within one of the levels of the namespace tree involved in the first, second, and third locks, as required by claim 10.

McClaughry et al. and Nitta et al. also do not disclose or suggest these features of claim 10.

For at least these additional reasons, Applicants submit that claim 10 is patentable over Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination.

Independent claims 13-15 and 17 recite features similar to, but possibly different in scope from, features recited in claim 1. Claims 13-15 and 17 are, therefore, patentable over Adya et al., McClaughry et al., and Nitta et al., whether taken alone or in any reasonable combination, for at least reasons similar to reasons given with regard to claim 1.

Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejection of claims 1-15 and 17 under 35 U.S.C. § 103 based on Adya et al., McClaughry et al., and Nitta et al.

CONCLUSION

In view of the foregoing amendments and remarks, Applicants respectfully request the Examiner's reconsideration of the application and the timely allowance of pending claims 1-17.

Applicants respectfully request that this Amendment under 37 C.F.R. § 1.116 be entered by the Examiner, placing claims 1-17 in condition for allowance. Applicants submit that the

proposed amendments do not raise new issues or necessitate the undertaking of any additional search of the art by the Examiner, since all of the elements and their relationships claimed were either earlier claimed or implied in the claims as examined. Therefore, this Amendment should allow for immediate action by the Examiner. Further, Applicants submit that the entry of this Amendment would place the application in better form for appeal, should the Examiner dispute the patentability of the pending claims.

As Applicants' remarks with respect to the Examiner's rejections overcome the rejections, Applicants' silence as to certain assertions by the Examiner in the Office Action or certain requirements that may be applicable to such rejections (e.g., whether a reference constitutes prior art, motivation to combine references, etc.) is not a concession by Applicants that such assertions are accurate or that such requirements have been met, and Applicants reserve the right to dispute these assertions/requirements in the future.

If the Examiner does not believe that all pending claims are now in condition for allowance, the Examiner is urged to contact the undersigned to expedite prosecution of this application.

To the extent necessary, a petition for an extension of time under 37 C.F.R. § 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account No. 50-1070 and please credit any excess fees to such deposit account.

Respectfully submitted,

HARRITY SNYDER, L.L.P.

By: /Paul A. Harrity/
Paul A. Harrity
Reg. No. 39,574

Date: September 15, 2006

11350 Random Hills Road
Suite 600
Fairfax, Virginia 22030
(571) 432-0800